Internet Draft
Expiration: May 2002
File: draft-braden-2level-signal-arch-00.txt

R. Braden
USC/ISI
B. Lindell
USC/ISI

A Two-Level Architecture for Internet Signaling

November 12, 2001

Status of Memo

Abstract

This memo suggests a two-level organization for Internet signaling protocols: a common lower layer called the Common Signaling Transport Protocol (CSTP), and an upper-level signaling protocol to implement the algorithms and data structures specific to a particular signaling application. CSTP features would include reliable delivery and soft state management.

1. Introduction

1.1 A Signaling Protocol Suite

Version 1 of RSVP was designed specifically to perform QoS signaling for Integrated Services [ISInt93]. However, many RSVP extensions have been developed or proposed to support a variety of other Internet signaling applications. These applications include:

- QoS setup across diff-serv clouds [intdiff00]
- Setting up MPLS paths with QoS [mpls00]
- Provisioning VPNs [aggr01]
- Configuring optical networks [optical00]
- QoS setup for PacketCable [PCQoS99]
- Active interest filtering for distributed simulation [AIF01].

With these extensions, RSVP Version 1 has in effect been expanded to define a suite of Internet signaling protocols. Basing all of these protocols on RSVP brings some unity that is highly desirable. For example, the signaling applications listed above benefit from RSVP's transport, routing, and soft-state mechanisms as well as from its strongly-typed encoding. Using a common protocol base also has benefits in design economy and documentation. On the other hand, the complexity of the resulting multi-featured RSVP implementations and the confusion of feature interactions are the source of considerable complexity and some confusion.

This memo proposes an approach to building an "Internet signaling protocol suite" (ISPS) with improved modularity. It falls short of a a more ambitious long-range goal, building the ISPS in a modular fashion out of common software building blocks. We don't yet know how to perform a modular decomposition for Internet signaling. This memo proposes a simpler step: the composition of Internet signaling protocols from two protocol levels: (1) a common lower level that performs transport-layer functions and (2) a set of upper-level signaling functions that are specific to

particular signaling applications.

In particular, we suggest a common lower-level protocol called CSTP ("Common Signaling Transport Protocol") to implement transport and soft-state functions. CSTP will support a suite of

higher-level "Application-Layer Signaling Protocols" (ALSPs). Each ALSP will implement the algorithms and data structures for a particular signaling task. CSTP together with the set of ALSPs will implement the Internet signaling protocol suite (ISPS). This modularity will allow the transport functions of CSTP to evolve more-or-less independently of the signaling applications.

We use the term "level" rather than "layer" for the ALSP/CSTP split, because traditional protocol layering implies an independence that will not exist in this case. It is true that ALSP and CSTP correspond roughly to application-layer and transport layer protocols, respectively; however, they are inter-dependent in ways that will be described below.

The conception behind this memo is not original. One of the advances in STream protocol II (ST-II) [RFC1191] over its predecessor ST was the explicit definition of a reliable hop-by-hop control sub-protocol called ST Control Message Protocol (SCMP). We believe that CSTP reflects some important advances over SCMP, for example soft state management.

This memo is an initial sketch of our two-level architecture for Internet signaling. This specification is incomplete in many respects, and since the ideas in this memo have not yet been subjected to either simulation or prototyping, it may very well include errors. However, we are offering a possible starting point for further protocol engineering.

The remainder of this section introduces some terminology and then summarizes the requirements on CSTP. Section 2 presents a general overview of CSTP, and Section 3 contains the beginning of a protocol specification. This memo makes many references to the RSVP Version 1 specifications [RFC2205, RFC2961]; the reader is assumed to be generally familiar with these RFCs. Of course, an ultimate specification for CSTP would have to be self-sufficient.

1.2 Terminology

We first introduce some useful terminology.

- Path-Directed Signaling

  Any protoocol that is used to set up state in network entities -- e.g., DHCP or IGMP -- might be called a "signaling protocol". However, our two-level ISPS architecture is intended to support RSVP-like signaling applications that install, modify, and remove state in routers and other entities along the path of some particular

  data flow. We call this paradigm "path-directed" signaling [Waypoint00]

  Path-directed signaling operates in the nodes along a data path between two (or more, for multicast) "signaling endpoints". This data path is the "signaled path", which has an initial node " p-src" and one or more terminal nodes "p-sink".

  For each signaling activity in a node along the signaled path, the directions "upstream" and "downstream" are defined relative to the data flow that defines the path. Thus, data flows downstream from p-src to p-sink, while signaling may occur in one or both directions. The P-src and p-sink nodes might be end systems that are the ultimate sources and destinations of the data packets that establish the path, or they might be intermediate nodes such as border routers or aggregation points or tunnel endpoints.

- CSTP Neighbors

  We define two CSTP-capable nodes as "neighbors" if they are connected by at least one path that includes no other CSTP-capable nodes. Neighbors that are directly connected, i.e., with no nodes intervening, are "direct neighbors". A CSTP-capable node may have at most one neighbor through each point-to-point interface, but it may have multiple neighbors through a broadcast or NBMA interface.

Signaling messages are generally sent hop-by-hop. Each hop is between neighbors, from an "h-src" (hop source) node to a neighbor node called "h-sink" (hop sink).

- SAPU

  A Signaling Application Protocol Unit (SAPU) is the basic transmission unit for signaling. A SAPU is derived from the signaled state in the h-src node and it is used to set, modify, or delete state in the h-sink node.

- Trigger, Refresh Messages

  A "trigger message" installs, modifies, or deletes signaled state, while a "refresh message" only refreshes existing state, i.e., prevents it from timing out.

1.3 CSTP Requirements

The partition of functionality between CSTP and ALSP is a tradeoff between generality and unity. A "thicker" CSTP level, i.e., one that has more function, would provide greater unity among signaling tasks. On the other hand, a "thicker" CSTP would also be less general and more likely to constrain the range of signaling protocols that can be achieved by any ALSP. This memo suggests a fairly "thin" CSTP, which includes a set of functions that are closely interlinked and that are generally useful for a broad range of signaling applications. For example, this CSTP will support signaling tasks that require simplex or full-duplex signaling, and it will support receiver- or sender-initiated signaling.

DISCUSSION

Suppose that the the current Version 1 RSVP functionality were to be mapped into a (CSTP, ALSP) pair (see Appendix A.) Neither RSVP's receiver-oriented operation nor its reservation styles [RFC2205] should appear in CSTP; these features would be implemented only in the RSVP-specific ALSP module.

The Common Signaling Transport Protocol (CSTP) should meet the following general objectives.

- Support for Path-Directed Signaling

  CSTP must support path-directed signaling, although some ALSPs may not make use of this capability. All end-to-end signaling semantics would be realized by the actions of an ALSP; CSTP itself does not have end-to-end semantics. CSTP handles only the dynamics of reliably transmitting signaling state between neighbors, h-src to h-sink, and of refreshing this as soft state.

  CSTP should not constrain the granularity of the data flow that defines a signaling path (although an ALSP might.) The flow granularity might range from micro-flows that are created by particular user applications to highly-aggregated flows.

- RSVP Version 1 Support

  It must be possible to support the functionality of any flavor of RSVP Version 1 using the combination of CSTP with an appropriate ALSP. We could therefore consider the (CSTP, ALSP) design as RSVP Version 2, although CSTP is deliberately

designed to be more general than a strictly RSVP-like protocol.

The CSTP design that is presented here would not directly interoperate with RSVP Version 1, due to differing packet formats. However, a signaling gateway could be developed to translate RSVP Version 1 signaling messages to and from (CSTP, ALSP) messages.

- Reliable Delivery of Signaling Messages

  Signaling operation must not be threatened by packet loss or reordering. Thus, CSTP must provide reliable delivery of trigger messages so that state can be reliably and promptly added, changed, and explicitly removed.

The division of function between CSTP and the ALSP will ease future modifications in the service model. For example, should it be decided that reliable delivery of refresh messages was desirable, this service could be added to CSTP with no changes to the ALSPs.

The reliable delivery algorithm must consider fragmentation. CSTP should be able to retransmit individual fragments of large SAPUs, to avoid bad error statistics when fragments are dropped. On the other hand, SAPUs larger than a very small number of MTUs are expected to be rare, so selective acknowledgments may not be very useful; a simple cumulative ACK should be sufficient.

DISCUSSION

The early design of RSVP Version 1 made the optimistic assumption that signaling traffic could be protected by QoS and that reordering would be rare. Experience later showed that these assumptions could be violated unacceptably often, so a reliable delivery mechanism [Refresh00] was pasted onto RSVP Version 1. Reliable delivery of trigger messages is a fundamental objective for CSTP, although a particular ALSP may choose to not use it.

- Ordered Delivery of SAPUs

The original RSVP v1 protocol spec [RFC2205] allowed network reordering of signaling packets to create significant (e.g., 30 second) periods of erroneous reservation. The addition of reliable delivery prevents this particular failure mode, but

it introduces the problem of delayed delivery of old duplicate packets. Therefore, the CSTP outlined in this memo includes a mechanism to ignore out-of-order trigger messages.

- Soft State Support

Even with reliable message delivery, there is cause for concern about the robustness with which unused state is

removed. In the somewhat chaotic multi-vendor environment of the Internet, it is unwise to assume error-free interoperation of many different implementations. Bugs that leave behind orphan state are particularly serious. We therefore include soft state -- removing state that is not periodically refreshed or explicitly torn down -- as a fundamental robustness mechanism of CSTP, although again a particular ALSP may choose to not use it.

- Fragmentation, Reassembly, and Bundling of SAPUs

  CSTP must be able to fragment and reassemble SAPUs that exceed one MTU.

  > We expect that elementary ISPS messages will be only a little bit larger than the corresponding RSVP Version 1 messages; the majority of SAPUs should be under 200 bytes. The addition of security credentials may lead to some SAPUs O(1000) bytes, but SAPUs signficantly larger than this are expected to be rare.

  Bundling -- carrying multiple small SAPUs in a single IP datagram -- may be desirable for performance, especially when there is a burst of signaling requests. Such a burst might be caused by a route change or by some external event. On the other hand, bundling can reduce the number of packets by only roughly a factor of 10. While this factor may be useful, it is not generally scalable. The bundling requirement may need further consideration.

- Full Internet-Layer Support

  CSTP should support the full range of Internet-layer protocols, including IPv4 and IPv6, unicast and multicast delivery, and IPSEC for data security.

- Partial Deployment

  It must be possible for signaling protocols supported by CSTP to operate correctly through CSTP-incapable nodes. This

requirement, together with support for path-directed

signaling, can be met by sending signaling messages downstream using the destination address of the data. Such messages will automatically be forwarded correctly through CSTP-incapable nodes. This mechanism in turn requires that each CSTP hop intercept signaling messages from the data stream [Waypoint00], process and perhaps modify them, and then forward them.

o Congestion Control

It would seem that the signaling protocol and the network configuration could ensure that signaling traffic will almost always be small relative to the data flow. However, all Internet traffic should be responsive to congestion.

Signaling data normally has the characteristics of packet video data: a long-lived (in fact, never-ending), somewhat bursty, stream of data. It should be possible to throttle back signaling bandwidth between a pair of nodes by slowing soft-state refreshes and by capping the rate of change of existing state, for example. In this regime, the techniques of TCP-friendly congestion control may be applicable to CSTP. However, bursts of trigger messages and retransmissions can also occur, so CSTP can also have TCP-like characteristics. Thus, reliable delivery introduces the need to dynamically compute the appropriate value for retransmission timers, and this computation must consider the round trip time (RTT) and network congestion.

The two-level signaling design centralizes all issues relating to the volume and timing of network signaling traffic within the common CSTP protocol. The CSTP module is in a position to perform complex scheduling of signaling message transmissions, taking into account the congestion at each target node and the signaling load. For example, CSTP might limit the rate of signaling traffic but still allow a burst of signaling traffic when a route changes. CSTP might be a good candidate for the "Congestion Manager" [CM01].

o Hop-by-hop integrity

Since the CSTP operates strictly hop/hop, it seems a natural place to implement (optional) hop-by-hop integrity, using the RSVP algorithms [Integrity00] for example.

All of these functions except the last are tightly coupled with each other, so they represent a logical set for CSTP to implement.

Integrity is included in the CSTP simply to provide common support for an important signaling component.

There are some common functions that do not seem to belong in CSTP. For example, it could be tempting to try to limit knowledge of unicast and multicast routing and addressing to the CSTP. Thus, the CSTP might provide Logical Interface Handles (LIHs) to support asymmetric paths [Section 3.3, RFC2205] as well as the Iapp logic that is required because of complexities of multicast routing [Section 3.9, RFC2205]. However, at present it appears that the ALSP must know about unicast or multicast routing and LIHs.

At this stage, the reader is likely to be wondering whether it would not be better to base CSTP on TCP, rather than building new TCP-like mechanisms for fragmentation/reassembly, reliable delivery, and congestion control. A virtue of the two-level architecture suggested in this memo is that a switch to CSTP over TCP would not change the ALSPs or the ALSP/CSTP interface.

2. Overview of CSTP

2.1 General

The two-level architecture will operate in the following general manner.

- To send a SAPU containing signaled state to a target neighbor node, the ALSP will issue a downcall to its local CSTP module. The local CSTP will encapsulate the SAPU in a signaling message and send it to the CSTP module in the target node. The destination address for this message may be p-dest (or the ultimate destination address if different from p-dest) for a downstream message, or it may be the IP address of the specific neighbor h-sink.

- The local CSTP will retransmit this message as necessary until an acknowledgment is received from the target CSTP.

- Once the transfer is acknowledged, the local CSTP will begin sending refresh messages for it (unless the message was deleting state). Note that the ALSP will not be involved in reliable delivery or refresh of this SAPU after the initial downcall.

The source node's ALSP module derives a SAPU from its stored state

and its CSTP carries the SAPU to the h-sink node, whose ALSP updates its stored state to reflect the SAPU. The ALSP is

responsible for maintaining consistent signaled state along the path. Thus, upon receiving a new or modified SAPU from a node h-src, an ALSP module may invoke its CSTP to forward appropriate SAPUs to other neighbors.

The information included in an SAPU is logically a (<key>, <value>) pair. The <key> part distinguishes the state specified by the <value> part from other state sent between the same pair of neighbors. However, the distinction between <key> and <value> within the SAPU is known only to the ALSP module; CSTP treats the SAPU as opaque.

CSTP messages may be sent as IP or UDP datagrams.

We believe that it is useful to retain the typed "object" syntax and the object encoding rules of RSVP Version 1, so that an SAPU will typically be a encoded as a sequence of elementary (type, length, value) triplets.

2.2 Reliable Delivery

The CSTP suggested here incorporates a version of the RSVP "refresh reduction" extensions [Refresh00] to provide reliable delivery of trigger messages and rejection of old duplicates. Note that the CSTP handles most or all of the event timing, so the ALSP can be event-driven by upcalls from the CSTP.

For reliable delivery, each trigger message includes a unique identifier, the SAPUid. The SAPUid is used as a handle on the SAPU that is known to the CSTP (as opposed to the <key>, buried within the SAPU, that the CSTP cannot see). A SAPUid is used for acknowledging receipt of a trigger-message SAPU and for efficiently refreshing the corresponding state.

EXAMPLE FROM RSVP V1

For the equivalent of an RSVP Resv message, the <key> part of the SAPU would consist of the SESSION and NHOP objects and perhaps (depending upon the STYLE) the FILTER_SPEC objects.

Other fields -- e.g., STYLE and FLOWSPEC -- wouild be in the <value> part. These complex rules on RSVP V1 <key>s would not be known by CSTP.

Note that some RSVP Version 1 objects may be mapped into the CSTP message header rather than included in a SAPU. For example, in RSVP the next/previous hop IP address is coupled with the Logical Interface Handle (LIH) in a single data object (NHOP/PHOP). Section 3 shows that the CSTP header includes the

next/previous hop IP address, but an ALSP would include the LIH in its SAPU only when it was needed by the particular signaling application.

CSTP transports SAPUs in DnSig (down-stream signaling) messages and UpSig (upstream signaling) messages. We use the term "xSig" to denote an elementary CSTP signaling message without specifying the direction.

A CSTP module must maintain state that lists the node's neighbors. This state includes the IP address of the neighbor and the local interface used to reach it. This per-neighbor state also includes two Boolean flags giving important properties of the neighbor, e.g., ISPS-capable and Direct-Neighbor. A node builds the neighbor list as a result of receiving CSTP messages. The neighbor list should be implemented as soft state that is deleted if it is not refreshed.

2.3 Example: Sending New State

Sending new signaled state involves the following sequence of steps. Some secondary parameters are omitted here for simplicity, but they appear in the more detailed spec in Section 3.

1. The local ALSP issues the following downcall to its CSTP, passing the new SAPU and a target IP address IP-target.

   SendNewSAPU(SAPU, IP-target, [OIf]) -> SAPUid

   For downstream transmission, IP-target will be either the target signaling destination address p-dest or the address h-sink of a neighbor. For upstream transmission, it must be

a neighbor address h-sink. The optional Outgoing InterFace (OIf) parameter is needed when IP-target is a multicast address.

The CSTP:

- o generates an SAPUid,
- o creates a local send state block,
- o builds and sends the trigger message:

  xSig(NEW, h-src, SAPUid, SAPU)

  to the IP-target address,

- o sets a retransmit timer,
- o and returns the SAPUid to the ALSP, which records this handle.

2. If the retransmit timer goes off before the NEW message is acknowledged, the local CSTP retransmits the trigger message. This is repeated until either an ACK is received or a limit is reached. In the latter case, the CSTP issues the upcall:

   SendFail(SAPUid, SAPU)

   and deletes the send state block.

3. Otherwise, when the CSTP receives a xSig(ACK, SAPUid) message, it stops retransmitting and starts sending periodic refresh messages to IP-target:

   xSig(REFRESH, h-src, SAPUid)

4. If the CSTP receives a xSig(NACK, SAPUid) message, it returns to step 2 to (re-)transmit the trigger message.

5. When the NEW message is received at the h-sink node that was implied or specified by IP-target, the remote CSTP:

- Creates a local receive state block,
- passes the SAPU to the remote ALSP via an upcall:

  RecvNewSAPU(SAPU, h-src)

- and returns an ACK message.

2.4 Ordered Delivery

Under soft-state signaling, old trigger messages should always be ignored. This can be accomplished by introducing a monotone-increasing sequence number in trigger messages.

Following the example of the Refresh Reduction extensions to RSVP v1 [Refresh00], we can overload the SAPUid to serve as a sequence number as well as a handle on reservation state. An h-src node would generate monotone increasing values for new SAPUids to be sent to a given h-sink. The h-sink node would then: (1) remember the largest SAPUid seen so far from h-src; (2) if a received SAPUid is larger, process that SAPU as a trigger message; (3) if the received SAPUid matches that of existing state from h-src,

treat the message as a refresh; (4) otherwise ignore the message and send a NACK.

It of course necessary to get the details of this mechanism right. When a node crashes and restarts, losing its state, some mechanism is required to reliably instruct its neighbors to reset their latest sequence numbers. When a route changes and a REFRESH message is answered with a NACK, h-src must send the new trigger message with a new SAPUid; h-src must also upcall to inform its ALSP that the SAPUid has changed for the existing state.

An alternative approach to ordered delivery would be to use the sequence number that is already present in the hop-by-hop cryptographic integrity check mechanism [Integrity00]. The integrity mechanism also includes a Challenge/Response mechanism to robustly (and securely) reset the sequence number in neighbors at startup.

3. CSTP Specification

3.1 Signaling Messages

The nine currently-defined CSTP message types are as follows. They are shown schematically in functional notation with the type as the first parameter.

xSig(NEW, h-src, SAPUid, SAPU, R)

xSig(MOD, h-src, SAPUid, SAPU, old-SAPUid, R)

xSig(TEAR, h-src, SAPUid)

xSig(REFRESH, h-src, SAPUid, R)

xSig(ACK, h-dest, SAPUid-list)

xSig(NACK, h-src, SAPUid)

xSig(EVENT, h-src, SAPUid, SAPU)

xSig(CHALLENGE, h-src, challenge-object)

xSig(RESPONSE, h-src, challenge-object)

Here:

- Every message contains the IP address of its originator, h-src. In most but not all cases this address is the same as

the source IP address of the ISPS packet. For simplicity we specify that h-src will always appear explicitly in a CSTP header. It is used to build neighbor state.

- R specifies the refresh time for the SAPU (see [RFC2205]).

- For the MOD message, the sending ALSP must ensure that the new SAPU with identifier SAPUid and the old SAPU with identifier old-SAPUid share the same <key> parts.

The EVENT message sends an SAPU reliably but does not continue to

refresh it. Thus, it represents a one-time signal or "event".

The CHALLENGE and RESPONSE messages are used to initialize the keyed hash integrity check [Integrity00]. The <challenge-object> is carried as a CSTP-level SAPU, which is a special case; all other SAPUs are opaque to CSTP and carried on behalf of an ALSP. <challenge-object> is defined in [Integrity00].

3.2 Header Format

In order to handle both fragmentation and bundling, the CSTP header could be divided into two parts: an outer header called the "B-header" ("B" for "bundle") and an inner header called the "M-header" ("M" for "message"). The basic CSTP packet format would be the sequence:

<B-header> <M-Header> [SAPU]

The optional SAPU is included if the CSTP header type in <M-header> calls for it. The other parameters shown in the previous section would be in <M-header> or <B-header>.

Multiple (<M-header>, [SAPU]) pairs, prefixed by a single <B-header>, can be sent in a single MTU-sized datagram; this is bundling. On the other hand, if the basic packet format is too large to fit into a single datagram, the (<M-header>, SAPU) pair can be fragmented into datagrams that are each prefixed with a <B-header>.

This header syntax can be summarized by the following BNF for a message:

<B-header> { <M-header-MF> <SAPU> }* <M-header> [<SAPU>]

The detailed engineering of the headers is being deferred for the present. However, in general the headers will contain the following information fields.

The B-header contains:

o The total length of the datagram in bytes

- A fragment offset and MF ("More Fragments") bit
- A checksum or keyed hash integrity object
- An ALSP identifier
- The R value for all messages bundled into this datagram.

The ALSP identifier is used to select the ALSP, and hence the signaling task, that is in use. We assume that there will be a simple registration space for ALSP identifiers. We believe this will work in practice, although the number of different ALSPs could be at worst 2**n, where n is the number of RSVP extensions. In practice, we expect only a few distinct combinations to exist.

The M-header contains:

- The length of the (<M-header>, [SAPU]) pair; this is the byte offset of the next M-header in the case of bundled messages.
- The CSTP message type for this message
- A list of zero or more SAPUids

If the CSTP message type calls for an SAPU, it follows immediately after the M-header. The first two bytes of the SAPU must be its length in bytes; otherwise, the SAPU format is entirely opaque to CSTP.

This account omits details like version numbers, IP address format specification, and flags.

3.3 State Diagram for CSTP

The stages in handling a particular SAPU can be summarized by the state diagram for the sending CSTP in h-src; it is shown in Figure 1. Here SendNewSAPU(), SendModSAPU(), and SendTearSAPU() represent down calls from the ALSP to the CSTP to install a new SAPU, modify an existing SAPU, or delete an SAPU, respectively. xSig() represents a CSTP message encapsulating an ALSP payload. TO-X refers to a retransmission timer firing, while TO-R and TO-T refer to refresh and state timeouts, respectively.

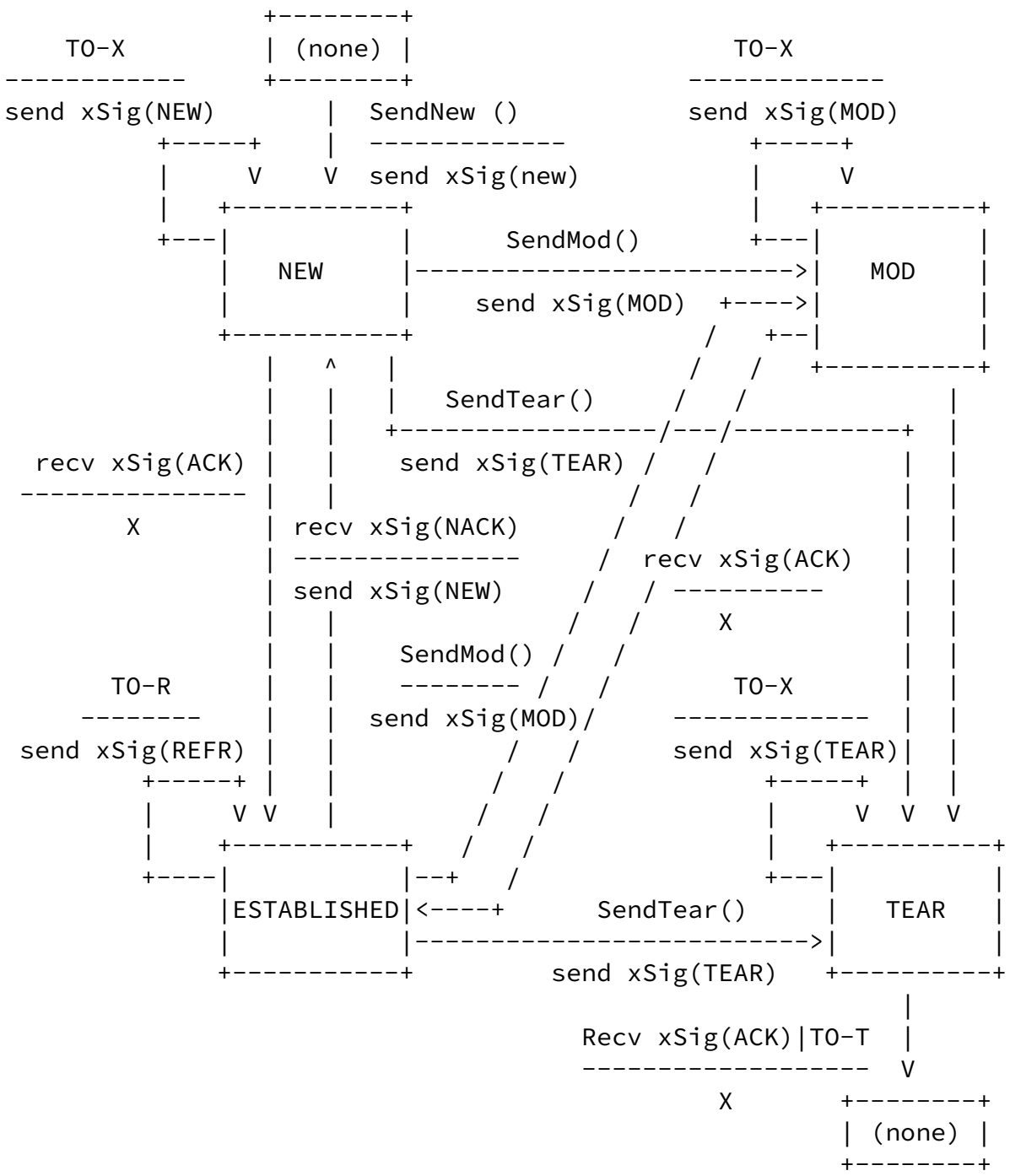

Figure 1: H-Src CSTP State Diagram

3.4 CSTP/ALSP Interface

This section defines a generic interface between CSTP and ALSP. For simplicity we assume that the two levels are distinct, sharing no data structures. This means that data structures must be passed across this interface by value and that the CSTP must keep a shadow copy of the SAPU state to be retransmitted. An actual implementation is likely to share data structures between the two levels to avoid this inefficiency. (An analogous relationship occurs between IP and TCP in most protocol implementations).

3.4.1 Down Calls

An ALSP issues the following downcalls to the CSTP.

o Send New SAPU

SendNewSAPU(SAPU, IP-target [, OIf], burst_flag)

-> SAPUid

This call sends the specified SAPU reliably to the neighbor specified or implied by address IP-target, and then allocates and returns a unique identifier SAPUid. If reliable delivery fails, CSTP issues an asynchronous SendFail() upcall to the ALSP. If the SAPU is delivered and acknowledged, CSTP then sends periodic soft-state refresh messages for it. CSTP continues the refresh autonomously until the ALSP makes a SendModSAPU() or sendTearSAPU() downcall for the same SAPUid.

If a route change later causes loss of state in a neighbor, CSTP will make a RegenSAPU() upcall to ask the ALSP to reconstruct the original SAPU, and then send this CSTP in a NEW trigger message containing a new SAPUid. The upcall will also transmit the revised SAPUid to the ALSP.

In the downstream direction, IP-target may be the signaling destination's IP address; the neighbor h-sink on

the path to IP-target will intercept and process the message. Otherwise, IP-target it must be the IP address of a neighbor (h-sink). For a multicast IP-target address, the caller may specify the outgoing interface OIf to be used.

In order to retransmit for reliable delivery, the CSTP may cache a copy of the SAPU. If an SAPU to be retransmitted

is not in the cache, the CSTP can issue a RegenSAPU() upcall to ask the ALSP to reconstruct the SAPU.

The burst_flag parameter is a boolean flag that can be used by the CSTP as a "hint" about when it can efficiently bundle a set of successive calls. When CSTP issues a burst of successive calls to SendNewSAPU(), all except the last should have this flag set to True. CSTP will make the decision about when to bundle. This allows the CSTP to avoid the introduction of substantial bundling delays.

- Send Modified SAPU

  SendModSAPU(mod-SAPUid, mod-SAPU, old-SAPUid, burst_flag)

  Modify an existing SAPU that had identifer old-SAPUid to be mod-SAPU with identifier mod-SAPU.

  Mod-SAPU will be reliably delivered and refreshed at the neighbor specified or implied by IP-target, or else CSTP will issue a SendFail(mod-SAPUid, reason) upcall to the ALSP.

- Remove Existing SAPU

  SendTearSAPU(SAPUid)

  Tear down the SAPU that corresponds to SAPUid.

- Send "Event"

  It is possible to send an SAPU reliably but not refresh it as soft state. Such a transmission is called an "event".

SendEventSAPU(SAPU, IP-target [, OIf], burst_flag)

This call is identical to SendNewSAPU(), except CSTP does not retain state after the transmission is acknowledged or times out.

- Send "Info" Message

  An SAPU can be sent with neither reliable delivery nor refreshing, i.e., sent as a datagram. This is called an "Information" or "Info" message.

  SendInfoSAPU(SAPU, IP-target [, OIf], burst_flag)

3.4.2 UPcalls to ALSP

The CSTP has the following upcalls to the ALSP.

- Notify of Send Failure

  SendFail(SAPUid, reason)

  Reports to ALSP that the SendNewSAPU() or SendModSAPU() operation failed for specified SAPUid.

- ALSP Receive New SAPU

  RecvNewSAPU(SAPU, h-src)

  A new SAPU has been received from the node whose IP address is h-src.

- ALSP Receive Modified SAPU

  RecvModSAPU(SAPU, h-src)

  An existing SAPU has been modified. Note that the ALSP can parse the SAPU and find the <key> in order to identify the older SAPU state to be replaced, so the SAPUid is not passed up.

Note also that the new/mod distinction here is not really needed; the ALSP will discover the status when it looks up the <key>. It is included in the interface only as a consistency check.

- ALSP Notified of Teardown Received

  RecvTearSAPU(SAPUid, h-src)

- Request ALSP to Regenerate SAPU

  RegenSAPU(SAPUid [, new-SAPUid]) -> SAPU

  Requests that the ALSP regenerate and return the SAPU corresponding to SAPUid. If the optional new-SAPUid parameter is present, the ALSP also uses it to replace SAPUid as its internal handle for this atom of signaled state.

APPENDIX A. RSVP Version 1 as an ALSP

To write an ALSP specification for the base Version 1 RSVP protocol of RFC 2205, we can adopt nearly all of RFC2205. This is largely because many of the issues handled by CSTP are dealt with in the Refresh Reduction extension document [Refresh00], not in RFC 2205. The Refresh Reduction document [Refresh00] would be entirely obsoleted by our ISPS proposal, although we have suggested adopting its basic concepts.

Looking at RFC 2205 in detail, we find the following.

- Section 1 of RFC 2205 would be little changed. This section discusses the objectives of RSVP and defines a session, a flowspec, a filterspec, receiver-initiated reservations, scope, reservation merging, and styles.

- Section 2 of RFC 2205 which describe the RSVP protocol mechanisms in general terms, would be changed only where it describes soft state and specific RSVP Version 1 message types. RSVP Version 1

message types would become a combination of SAPU type and CSTP message types, as shown in the table below. Note that a few of the RSVP Version 1 message types, e.g., Bundle, simply disappear into mechanisms included in CSTP.

o Section 3 of RFC 2205 contains the functional specification of RSVP Version 1, and section 3.1 defines RSVP Version 1 message syntax and semantics. Each <xxx Message> definition that maps into ISPS becomes a <yyy SAPU> definition. The Common Header is replaced by an SAPU header that contains only a length and an SAPU type. The INTEGRITY object is omitted since it will now appear in the CSTP header. Otherwise, Section 3.1 would be unchanged.

o Some discussion would be required of exactly how the RSVP ALSP should invoke the downcalls to CSTP and the upcalls from CSTP.

The message types of RSVP Version 1 will be mapped as follows, using the ISPS design of this memo.

RSVP Version 1 Message Type	SAPU Type	CSTP Message Type
Path	Path	NEW or MOD
Resv	Resv	NEW or MOD
Srefresh	Path or Resv	REFRESH
ACK	Path or Resv	ACK or NACK
PathTear	Path	TEAR
ResvTear	Resv	TEAR

PathErr	PathErr	EVENT
ResvErr, ResvConf	ResvErr	EVENT
DREQ	DiagReq	EVENT
DREP	DiagRep	EVENT
Integrity Challenge	(none)	CHALLENGE
Integrity Response	(none)	RESPONSE
Bundle	(none)	(CSTP header)
ResvTearConf ??		

APPENDIX B. OPEN ISSUES

The following issues were left unresolved in the CSTP specification in this document.

- Should the spec support optional modes of unreliable delivery but with refresh for NEW or MOD, or optional unreliable delivery for TEAR?
- Should there be an explicit upstream/downstream indicator in the CSTP header?
- Is MOD logically necessary, and is it useful?
- Are there important efficiency issues in not piggy-backing ACKs, NACKs upon existing messages?

Security Considerations

The CSTP protocol introduced in this document may support hop-by-hop integrity using the algorithms of RSVP version 1 [Integrity00]. Incorporation of the COPS mechanisms for secure end-to-end authentication and access control of signaling is a matter for further study.

References

[aggr01] Baker, F. et. al., "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001.

[AIF01] Keaton, M., Lindell, R., Braden, R., and S. Zabele, "Active Multicast Information Dissemination", submitted to conference, April 2001.

[CM01] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.

[intdiff00] Bernet, Y. et al, "A Framework for Integrated Services Operation over Diffserv Networks", RFC 2998, November 2000.

[Integrity00] Baker, F., Lindell, R., and M. Talwar, "RSVP Cryptographic Authentication", RSVP 2747, January 2000. 1996.

[ISInt93] Braden, R., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

[ISrsvp96] Wroclawski, J., "The Use of RSVP with Integrated Services", RFC 2210, September 1997.

[mpls00] Swallow, G., et al, "RSVP-TE: Extensions to RSVP for LSP Tunnels", <draft-ietf-mpls-rsvp-lsp-tunnel-09.txt>, IETF, Sept 2001.

[optical00] Rajagopalan, B., "LMP, LDP and RSVP Extensions for Optical UNI Signaling", <draft-bala-uni-signaling-extensions-00.txt>, IETF, October 2001.

[PCQoS99] "PacketCable(tm) Dynamic Quality-of-Service Specification", PKT-SP-DQOS-I01-991201, Cable Television Laboratories, Inc., 1999.

[Refresh00] Berger, L., et. al., "RSVP Refresh Overhead Reduction Extensions", <draft-ietf-rsvp-refresh-reduct-05.txt>, IETF, June 2000.

[RFC2205] Braden., R. Ed., et. al., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[Waypoint00] The path-oriented concept was explored in an expired Internet Draft: Lindell, B., "Waypoint -- A Path Oriented Delivery Mechanism for IP based Control, Measurement, and Signaling Protocols"", <draft-lindell-waypoint-00.txt>, IETF, November 2000.

Authors' Addresses

Bob Braden
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Phone: (310) 448-9173
EMail: Braden@ISI.EDU

Bob Lindell
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Phone: (310) 448 8727
EMail: Lindell@ISI.EDU